ChinaXiv合作期刊

RESEARCH PAPER

Probabilistic Tractable Models in Mixed Discrete-Continuous Domains

Andreas Bueff[1†], Stefanie Speichert[1] & Vaishak Belle[1,2]

[1]School of Informatics, University of Edinburgh, Old College, South Bridge, Edinburgh, EH8 9YL, UK

[2]Alan Turing Institute, British Library, 96 Euston Road, London NW1 2DB, UK

Keywords: Graphical models; Tractable inference; Hybrid domains; Weighted model integration

Citation: Bueff, A., Speichert, S., Belle, V.: Probabilistic tractable models in mixed discrete-continuous domains. Data Intelligence 3(2), 228-260 (2021). doi: 10.1162/dint_a_00064

Received: July 1, 2020; Revised: September 27, 2020; Accepted: October 14, 2020

chinaXiv:202211.00326v1

ABSTRACT

We study the problem of the unsupervised learning of graphical models in mixed discrete-continuous domains. The problem of unsupervised learning of such models in discrete domains alone is notoriously challenging, compounded by the fact that inference is computationally demanding. The situation is generally believed to be significantly worse in discrete-continuous domains: estimating the unknown probability distribution of given samples is often limited in practice to a handful of parametric forms, and in addition to that, computing conditional queries need to carefully handle low-probability regions in safety-critical applications. In recent years, the regime of tractable learning has emerged, which attempts to learn a graphical model that permits efficient inference. Most of the results in this regime are based on arithmetic circuits, for which inference is linear in the size of the obtained circuit. In this work, we show how, with minimal modifications, such regimes can be generalized by leveraging efficient density estimation schemes based on piecewise polynomial approximations. Our framework is realized on a recent computational abstraction that permits efficient inference for a range of queries in the underlying language. Our empirical results show that our approach is effective, and allows a study of the trade-off between the granularity of the learned model and its predictive power.

† Corresponding author: Andreas Bueff (Email: andreas.bueff@ed.ac.uk; ORCID: 0000-0002-7538-0699).

ChinaXiv合作期刊

chinaXiv:202211.00326v1

1. INTRODUCTION

Probabilistic representations, such as Bayesian and Markov networks, are fundamental to statistical machine learning and have applications in a range of domains, including biology, physics and robotics [1]. The attractiveness of such networks is that they can express probabilistic dependencies in a compact manner. Unfortunately, exact inference in these representations is intractable [2, 3]. Naturally, in the era of big data, there is also enormous interest in learning such structures directly from data. However, owing to the intractability of inference, learning also becomes challenging, since learning typically uses inference as a sub-routine [4], and moreover, even if such a representation is learned, the prediction will suffer because inference has to be approximated.

Tractable learning is a powerful new paradigm that attempts to learn representations that support efficient probabilistic querying. Much of the initial work focused on low tree-width models [5], but later, building on properties such as local structure [6], data structures such as arithmetic circuits (ACs) emerged. These circuit learners can also represent high tree-width models and enable exact inference for a range of queries in time polynomial in the circuit size. Sum-product networks (SPNs) [7] are instances of ACs with an elegant recursive structure – essentially, an SPN is a weighted sum of products of SPNs, and the base case is a leaf node denoting a tractable probability distribution (e.g., a univariate Bernoulli distribution). In so much as deep learning models can be understood as graphical models with multiple hidden variables, SPNs can be seen as a tractable deep architecture. Standard deep architectures rely on many layers of hidden variables for greater representational power, but inference with even a single layer is generally intractable, with additional layers exacerbating the problem. Tractable architectures, such as SPNs, have the advantage over standard deep learning methods in that they can learn unsupervised and they have full probabilistic semantics where inference is guaranteed to be tractable [8]. Of course, learning the architecture of standard deep models is very challenging [9], and in contrast, SPNs, by their very design, offer a reliable structure learning paradigm. While it is possible to specify SPNs by hand, weight learning is additionally required to obtain a probability distribution, but also the specification of SPNs has to obey conditions of completeness and decomposability, all of which makes structure learning an obvious choice. Since SPNs were introduced, a number of structure learning frameworks have been developed for those and related data structures, e.g., [10, 11, 12].

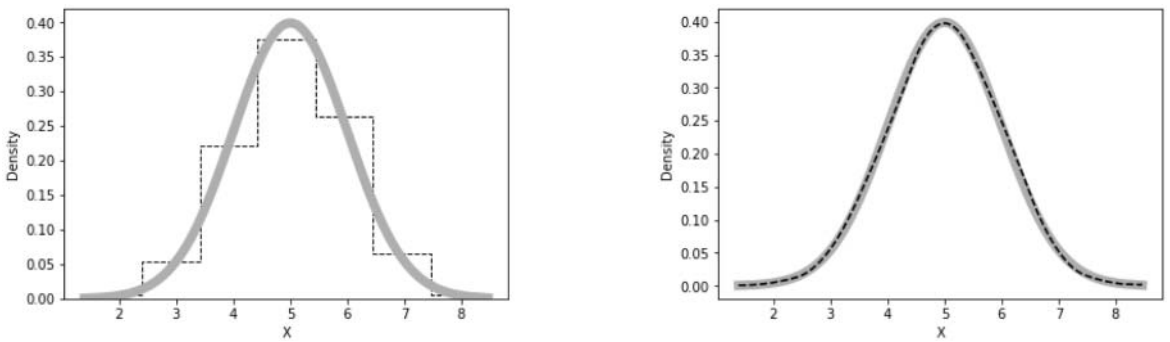
In this work, we study the problem of the unsupervised learning of tractable graphical models in mixed discrete-continuous domains. In general, tractability and learning issues aside, even in the inference context, the situation is generally believed to be significantly harder in discrete-continuous domains: estimating the unknown probability distribution of given samples is often limited in practice to a handful of parametric forms, and in addition to that, computing conditional queries needs to carefully handle low-probability regions in safety-critical applications, such as autonomous vehicles and health monitoring. Techniques based on Gaussian, conditional linear Gaussian [13] and mixing exponential families [14] are most common, for example.

To address the problem, then, we would first need a framework where it becomes possible to reason about continuous and mixed discrete-continuous event spaces in a general manner. With discrete models,

ChinaXiv合作期刊

chinaXiv:202211.00326v1

weighted model counting (WMC) has emerged as an assembly language for state-of-the-art reasoning in Bayesian networks, factor graphs, probabilistic programs, and probabilistic databases [6, 15]. Exact WMC solvers are able to handle low-probability observations, and the query language can support arbitrary propositional formulas over logical connectives, including disjunctions, which make them particularly useful in the presence of logical soft and hard constraints. However, WMC is limited to discrete finite-outcome distributions only, and little was understood about whether a suitable extension exists for continuous and discrete-continuous random variables, until recently. The framework of weighted model integration (WMI) [16] extended the usual WMC setting by allowing real-valued variables over symbolic weight functions, as opposed to purely numeric weights in WMC. These symbolic weights can take the form of piecewise polynomials, for example, and thus, WMI appeals to the emerging interest in density estimation by piecewise polynomial approximations [17, 18, 19, 20]. Essentially, these approximations can be made arbitrarily close to exponential families, among others including log-concave distributions, mixtures of Gaussians and Poisson Binomial distributions [21]. WMI additionally enables efficient integration [22] and recent WMI solvers [23, 24] are maturing quickly in the sense of attempting to leverage the powerful strategies already used in WMC. In essence, WMI is a strict generalization of WMC [16]; for example, just as inference in discrete graphical models reduces to WMC [2, 6], obtained from the sum of products of atoms' weights in a propositional model, inference in hybrid graphical models reduces to the WMI, obtained by integrating the product of atoms' densities in a first-order model. For example, a Gaussian distribution $\mathcal{N}(5,1)$ for a random variable x can be represented using (truncated) models seen in Figure 1.



(a) Constant Approximation (b) Third Order Polynomial

Figure 1. Comparison of piecewise constant *vs* polynomial using a 7-piece function.

As a computational strategy, what makes WMI particularly effective is that: (a) the intervals can be understood as (and mapped to) propositions [16], known in the literature as *propositional abstraction* [23], allowing us to piggyback on any propositional solver, and (b) WMI admits complex interval queries, such as $\Pr(x > 1{:}5 \mid y < 2)$ where x, y are random variables, yielding a powerful query interface. Notice here that the intervals in the queries do not have to correspond to the intervals used for specifying the distribution, and can overlap and subsume them arbitrarily.

To leverage WMI and piecewise polynomials more generally with structure learning, consider that the base case for SPNs is a tractable distribution, and so, SPNs are essentially a schema for composing tractable

chinaXiv:202211.00326v1

distributions. Although much of the literature focuses on univariate Bernoulli distributions [10, 12], continuous models can be considered too, which makes SPNs very appealing because real-world data are often hybrid, with discrete, categorical and continuous entities. Nonetheless, this rests on the assumption of having a tractable query interface to the underlying continuous and hybrid distributions. In [11], for example, the leaf nodes are assumed to be drawn from a Gaussian, and similarly, and [25, 26] assume the data are drawn from other families with a known parametric form.

In this work, we show how, with minimal modifications, structure learning can be generalized by leveraging efficient density estimation schemes based on piecewise polynomial approximations. Although SPN integration with parametric and non-parametric models is receiving increased attention, to the best of our knowledge, we are the first to achieve full integration of WMI and its powerful query mechanism, together with tractability guarantees regarding exact integration whilst achieving a clean separation of circuit learning from polynomial weight learning. In particular, owing to the tractability of integration over polynomial densities [27], inference becomes tractable, and can be computed by doing a few passes over the network. Moreover, among other things, the learning component allows us to induce both parametric and non-parametric models and their mixtures, including models such as Figure 1 and Figure 2, but with no prior knowledge of the true distribution, all of which are further composed over hidden layers in a deep probabilistic architecture. In other words, leveraging SPNs enables a fully *unsupervised* learning regime for hybrid data with WMI acting as an underlying inference framework. From a representational viewpoint, our framework allows the modeller to study the granularity of the continuous distributions (in terms of the number of piecewise components and the degree of the polynomial fit), and determine the empirical trade-off between accuracy and model size/interpretability. By lifting the propositional abstraction strategy of WMI, the framework is also instantiated in a manner that allows us to use any SPN learner in principle. To reiterate, to the best of our knowledge, this is a first attempt to combine WMI and circuit learning in their full generality, with an interface for complex interval queries. We will define this precisely in Section 3, but briefly the idea is that we can handle conjunctions and disjunctions of probabilistic queries, in the sense that these queries involve interval formulas but these intervals are not required to be mentioned in the specification of the distributions in an SPN. As a contrasting example, say we have the learning problem of credit risk. A deep neural network (DNN) would learn to classify an input customer as being either *BAD* for a customer potentially reneging on paying back a bank load or *GOOD* if the customer is likely to pay back a loan. With standard DNNs inference would be on the classification as $\Pr(\texttt{class} = \mathit{BAD} \mid X)$, where X is a vector of customer features (`age`, `job`, `creditamount`, `savingsaccount`, etc.). Inference requires X to have instantiated values but does not allow for more nuanced questions. In contrast with SPNs, specifically with our integration of WMI, we can now perform inference such as with $Pr(\texttt{class} = \mathit{BAD} \mid \texttt{job} = \mathit{Unemployed} \wedge 7{,}500 < \texttt{creditamount} < 9{,}000)$ which allows for performing inference on discrete and continuous features with specified ranges with respect to the continuous features. As we will see, handling such queries requires multiple passes over the SPN. The code for the framework is available on FigShare[①].

[①] https://doi.org/10.6084/m9.figshare.13012262.v1.

ChinaXiv合作期刊

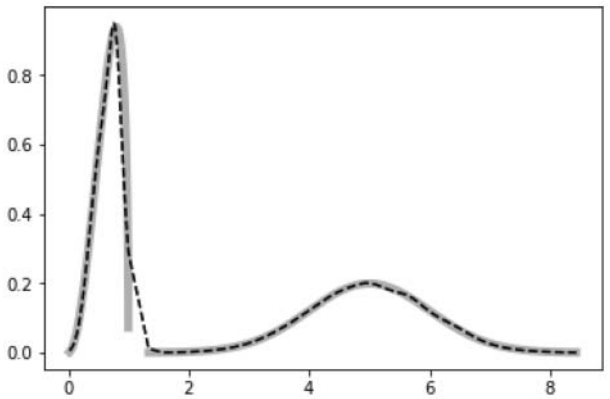


Figure 2. The piecewise polynomial approximate algorithm of a mixture of a Gaussian and Beta distribution.

This article is organized as follows. We discuss related work followed by the formal foundations for our work and then turn to our approach that integrates WMI and SPNs. We then discuss our interface for complex interval queries. We turn to empirical evaluations after that and finally conclude.

2. RELATED WORK

In addition to the related efforts we mentioned above, we remark that in a recent and independent effort[②], [28] introduces so-called Mixed SPNs (MSPNs). Like our work, they are motivated by piecewise approximations for learning from hybrid data. They propose a decomposition and conditioning approach for such SPNs employing the Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient [29]. While close in spirit, there are several significant differences to our work. Firstly, although their proposal is motivated by piecewise approximations, they focus on the learning of piecewise constant or linear models. From a representational viewpoint, this seems like a shortcoming, because in principle these weight functions can be effectively mapped to propositional SPNs with numeric weights, which remove much of the complexity, and hence appeal when finer polynomial density approximations are needed[③]. Secondly, the realization of an appropriate query interface that dynamically handles arbitrarily complex interval queries is not a focus of the paper – this is a critical feature with regard to our model as we take advantage of WMI. From a computational viewpoint, such a realization requires, as we show, subtle decompositions and multiple traversals to compute the resulting probability. To reiterate, what we strive for here is a full integration of SPNs and WMI. Thirdly, and perhaps most significantly, our approach neatly separates the propositional layer from the continuous aspects (including the integration) through pre-processing the data, which makes our framework generic, and applicable to any SPN learner. Finally, in our empirical evaluations,

[②] A preliminary version of our work was online on July 2018: https://arxiv.org/abs/1807.05464.

[③] It is possible that learning polynomials of arbitrary degree over intervals of arbitrary granularity are compatible with their approach, but without a discussion on the trade-off between accuracy and representation, and an algorithm for learning such polynomials, it is hard to assess such potential extensions.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

we show that our approach significantly outperforms [28] on almost all data sets in terms of the average test log-likelihood measure. On the other hand, it is argued in [28] that the decomposition technique naturally handles random variables of unknown type, whereas we treat all real-valued variables via intervalization and piecewise polynomial approximations. So it may be fruitful to combine the strengths of both Renyi decomposition of variables of unknown type with our handling of continuous variables of unknown distribution, which we leave for future research.

Regarding inference and MSPNs, a later paper [30] expands MSPNs to include an inference specifically geared for Approximate Query Processing (AQP). AQP aims to deliver an estimate of the query given a fixed time-bound. Kulessa et al. proposed two strategies for querying, and one based on likelihood inference on the model and the other based on sampling. The differences between [30] and our approach are notably similar to the differences our approach had with [28], that is, treating linear piecewise approximations as constants, and not allowing user-defined subinterval queries. Nonetheless, given the focus on SQL queries, the nature of querying is far broader than our approach. We also note the inclusion of sampling is a unique approach which holds merit given the broad set of user-defined SQL queries allowed, and we observe that the inclusion of our methods of complex querying would be a constructive addition to their SQL syntax. We elaborate further in Section 5.

Another body of work released after our initial findings also addresses inference concerning MSPNs [31]. Both our work and [31] use a similar separation of structure learning with distribution modeling at the leaf nodes. [31] relies on MSPNs to derive the structure of the data and then use Gibbs sampling in conjunction with a likelihood dictionary (e.g., Gaussian, Gamma, Exponential distributions) to approximate variable distributions at the leaf nodes (parameters for parametric models). This approach replaces the linear piecewise polynomials from [28] and uses a Bayesian inference approach with a Gibbs sampling scheme. They can perform interval querying on continuous variables that are similar to our approach (complex querying), as well as learn on missing data which we do not handle. Our approach, however, is not reliant on a likelihood dictionary of parametric distributions nor sampling for parametric parameters. As will be explained in Section 4, our model is more aligned with the general tractable inference mechanism of SPNs with distribution modeling handled by piecewise polynomial learning on the data. It would be interesting to combine our piecewise polynomial learning modeling method with the likelihood dictionary and explore a sampling approach in performing inference for future endeavors. Nonetheless, we note that we perform exact inference in our current framework, which we think fits well with the tractable learning paradigm of learning structures that permit efficient exact inference. Leveraging a sampling-based method would have to be approached carefully, and the merits and demerits evaluated thoroughly.

Outside of the above lines of research, research in learning in hybrid domains has been primarily limited to well-known parametric families. For example, [13] focuses on conditional linear Gaussian models, and [14] focuses on mixing exponential families. This is perhaps not surprising, because inference in hybrid domains has been primarily limited to approximate computations, e.g., [32], and/or Gaussian models [33]. In a similar spirit, approaches such as [34, 35] consider learning of complex symbolic constraints by assuming base distributions as being either Gaussian or a softmax equality. Another inference direction for

chinaXiv:202211.00326v1

ChinaXiv合作期刊

hybrid domains was done by [36], where SPNs interchange integrals with sum nodes to evaluate on continuous features that are provided with arbitrary input distributions. While that is a novel approach, we maintain an integration scheme at the leaf nodes to handle arbitrary interval queries.

A particular body work which utilizes parametric families for modelling hybrid distributions is Automatic Bayesian Density Analysis (ABDA) [31], which uses the SPN architecture to cluster feature distributions for mixture modelling, and SPN inference to impute missing values and for handling outlying values in the data. ABDA forgoes the piecewise linear approach of MSPNs and our piecewise polynomial method, as such we include comparisons of ABDA on hybrid data sets with our method in the experiments in Section 6. We also include experimental comparisons with Bayesian Sum-Product networks (BSPNs) which applies a Bayesian approach to structure learning by focusing on deriving so-called scope functions to formulate joint Bayesian functions over the structure and parameters of a SPN [37]. From our empirical evaluation we find our approach succeeds in beating both ABDA and BSPNs.

While our research focuses on structure learning, we mention an interesting framework which does away with SPN structure learning by constructing random region graphs populated with SPN nodes, so called random and tensorized SPNs (RAT-SPNs) [38]. A key aspect of RAT-SPNs is that by training the model discriminatively, it yields a classier on par with deep architectures. As the learning objective of a RAT-SPN can be supervised as opposed to unsupervised, a future direction of research could be the reframing of our learning method to be supervised by incorporating the discriminative training framework of RAT-SPNs. Another future research direction would be investigation into Einsum Networks (EiNet), which computes SPN sum and product nodes on the same topological layer via a single monolithic einsum operation [39]. Incorporating EiNet operations could lead to scaling up of large data set modelling with respect to our method.

From an inference viewpoint, WMI serves as a computational abstraction for exact and approximate inference [16, 40] based on piecewise polynomial approximations. It is thus different in spirit to variational methods and analytic closed-form solutions for parametric families. We refer readers to [16] for discussions. WMI has enjoyed considerable advances in recent years [41, 42], and in that sense, our framework might be the starting point for closing the gap between inference and learning.

chinaXiv:202211.00326v1

3. INFERENCE FRAMEWORK AND GRAPH ARCHITECTURE

This section presents the core components of our methodology by explaining WMI and SPNs. Emphasis is given to WMI's relationship with complex queries, including a formal definition of complex queries. We also provide an illustration of the probabilistic modelling capabilities of SPNs, as well as basic notation used throughout.

3.1 Weighted Model Integration

Given a propositional formula Δ, the task of model counting is to compute the total number of satisfying assignments for Δ. Given an assignment (model or world) M, we write $M \models \Delta$ to denote *satisfaction*. For

ChinaXiv合作期刊

example, $p \vee q$ has 3 models, and a satisfying ratio of 3/4. Weighted model counting (WMC) is its extension that additionally accords weights to the models [6]. Models are usually accorded weights by computing the product of weighted literals: that is, assuming w maps literals in Δ to positive reals, we define:

$$WMC(\Delta, w) = \sum_{M \models \Delta} \prod_{l \in M} w(l) \tag{1}$$

where the sum ranges over propositional models, and $l \in M$ denotes the literals true at the model M. For example, suppose p and q are accorded a weight of .6 and .3, respectively, with the understanding that the weights of a negated atom $\neg a = 1 - w(a)$. Then, the weight accorded to $[p = 1, q = 1]$ would be .18, and $WMC(p \vee q, w) = .18 + .42 + .12 = .72$.

WMC is a state-of-the-art exact inference scheme, and owing to its generality, inference in a number of formalisms, such as relational Bayesian networks and probabilistic programs [15], are encoded as WMC tasks. Given a formula Δ, a weight function w, a query q and evidence e, probabilities are computed by means of the expression: $\Pr(q \mid e, \Delta) = WMC(\Delta \wedge q \wedge e, w)/WMC(\Delta \wedge e, w)$.

Due to the propositional setting, however, WMC is limited to finite domain discrete random variables, which have spurred considerable interest in generalizing WMC to continuous and hybrid distributions [16, 18, 43]. Weighted model integration (WMI) is a computational abstraction for computing probabilities with continuous and mixed discrete-continuous distributions [16]. The key idea is to additionally consider inequality atoms, such as $0 \leq x \leq 10$, along with possibly non-numeric weights, such as x^2. The intuition is to let the inequality atom denote an interval of the domain of a density function, and let the weight define the function for that interval[④]. Formally, WMC was based on weighted propositional knowledge bases, and leverages SAT (propositional satisfiability) technology, and by extension, WMI is based on weighted linear arithmetic theories and leverages satisfiability modulo theory (SMT) technology [44].

Given such a knowledge base, the weighted model integration (WMI) is obtained from the model count of the theory together with a volume computation for the intervals. Formally:

$$WMI(\Delta, w) = \sum_{M \models \Delta^{-}} \int_{(l^{+} : l \in M)} \prod_{l \in M} w(l) \tag{2}$$

where Δ^{-} denotes a *propositional abstraction*, based on a mapping from linear arithmetic expressions to propositional atoms, and Δ^{+} denotes a *refinement*, where the atoms are mapped back to their linear arithmetic expressions. For example, $(0 \leq x \leq 10) \vee q$ on abstraction yields $p \vee q$, where p is a fresh atom chosen so as not to conflict with any of the existing atoms in the theory, which has a model count of 3 on abstraction.

④ Oblique supports such as $(x > y)$ or $(x - y > 5)$ may also be considered [45], but since we will be interested in weighted mixtures of univariate distributions, we will direct our attention to intervals here. Intervals can also be used to define many classes of multivariate distributions via hyper cubes [45]. These are intervals extended to multiple dimensions.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

Example 1 [16]

Suppose Δ *is the following formula:* $(0 \le x \le 10) \vee q$, *where* p *is the propositional abstraction for* $(0 \le x \le 10)$. *We also suppose that the weights are given with* $w(p) = x^2$ *and* $w(q) = 0.3$, *and the weights for negated atoms are* $w(\neg p) = 1$ and $w(\neg q) = 0$, *respectively. There are three models for* Δ^{-}:

1. $M = \{p, \neg q\}$: *Since* $w(\neg q) = 0$, *by definition we have*

$$\int_{(0\le x\le 10)} w(p)\cdot w(\neg q) = [\frac{x^3}{3}\cdot 0]_0^{10} = 0$$

2. $M = \{\neg p, q\} : \int_{(0\le x\le 10)} 1\cdot 0.3dx = [0.3x]_0^{10} = 3$

3. $M = \{p, q\} : \int_{(0\le x\le 10)} x^2\cdot 0.3dx = [0.1x^3]_0^{10} = 100$

Thus $WMI(\Delta, w) = 100 + 3 + 0 = 103$.

To understand the intuition here, consider that in the one-variable setting, we are assuming a density function $f : \mathbb{R} \to \mathbb{R}$ of the following form [45]:

$$f(x) = \begin{cases} a_{0i} + a_{1i}x + \cdots + a_{ni}x^n & \text{for } x \in A_i, i = 1, \cdots, k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where A_1, …, A_k are disjoint intervals in $\mathbb{R}$ that do not depend on x and a_{0i}, …, a_{ni} are constants for all i. We will say that f is a k–piece n–degree piecewise polynomial density approximation.

This is then encoded as a WMI atom A_i (e.g., $x \in [\alpha, \beta]$) with weight $w(A_i) = a_{0i} + \ldots + a_{ni}x^n$. For example, suppose we also have the query atom $\alpha \le x \le \beta$ and $e = true$. (We treat the weights of query atoms and their negations as being 1.) On abstraction, we would use a proposition p to stand for the interval A_i. Clearly, calculating the probability of the interval simply amounts to calculating the area under the curve given by the polynomial $w(A_i)$.

$$\Pr(x \in [\alpha, \beta]) = \int_{\alpha}^{\beta}(a_{0i} + \cdots + a_{ni}x^n)dx \tag{4}$$

This can also be approached as:

$$\Pr(p) = \int_{p^+} w(p)dp = \int_{x\in[\alpha,\beta]}(a_{0i} + \cdots + a_{ni}x^n)dx \tag{5}$$

Naturally, if the query refers to intervals not appearing in the specification, we have to decompose the problem. We will return to this point in more detail later (Section 5), but the rough idea is that given two pieces of the density function f, say $A_1 = \alpha_1 \le x \le \beta_1$ and $A_2 = \beta_1 \le x \le \beta_2$ with weights $P_1(x)$ and $P_2(x)$, clearly the probability of $x \in [\alpha^*, \beta^*]$ where $\alpha_1 < \alpha^* < \beta_1$ and $\beta_1 < \beta^* < \beta_2$ would be obtained as:

$$\Pr(x \in [\alpha^*, \beta^*]) = \int_{\alpha^*}^{\beta_1} P_1(x)dx + \int_{\beta_1}^{\beta^*} P_2(x)dx \tag{6}$$

While the mathematical treatment is immediate, computing these probabilities dynamically requires, as we will see, multiple passes over the SPN and so we will refer to such queries as *complex queries*.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

Definition 1 ***Complex Queries:*** *Given a k–piece n-degree piecewise polynomial density function f for r.v. x, defined over intervals A_i, …, A_k, we define a complex query as an interval query $q{:}\alpha^* \leq x \leq \beta^*$ where:*

1). *α^* or β^* is not one of the extreme points in A_1, …, A_k.*
2). *given two (not necessarily complex) queries q and q', we are interested in the probability of $q \circ q'$,* where $\circ \in \{\wedge, \vee\}$ or the probability of $\neg q$.

In general, the WMI apparatus is very flexible and on top of complex querying with continuous variables, we can also combine complex querying with Boolean variables.

3.1.1 Weighted Model Learning

Weight learning in the past [16] has focused on *piecewise-constant density approximations* where features are linear arithmetic sentences and weights are constants. This was addressed as follows: given a formula f_i, we wish to ensure that $\mathbb{E}_D[f_i] = \mathbb{E}_w[f_i]$ by *maximum likelihood estimation* (MLE) with weights $\mathbf{w} = (w_1, \ldots, w_k)$ and data D. This states that the empirical expectation of f_i in D (for example the count) equals its expectation according to the current parameterization.

More precisely given an SMT theory Δ, the empirical expectation of a formula $\alpha \in \Delta$ can be obtained by calculating:

$$\mathbb{E}_D[\alpha] = size(d \mid d \models \alpha \text{ and } d \in D) \,/\, size(D) \tag{7}$$

This simply counts the items in D which are true for α. Given a weight function w, the expectation of $\alpha \in \Delta$ wrt w is given by:

$$\mathbb{E}_w[\alpha] = WMI(\alpha, w) \,/\, Z \tag{8}$$

Here Z is a *normalization* constant, also referred to as the *partition function*. In our case, the partition function equals the integral of weights of all models of Δ, thus $Z = WMI(\Delta, w)$. Based on this formulation, standard iterative methods for convex optimization can be used for weight learning [16]. In our setting integration with circuit learning removes the burden of parameter learning from WMI, and moreover, we are able to handle piecewise polynomials density approximations.

3.2 Sum-Product Networks

SPNs are rooted acyclic graphs whose internal nodes are sums and products, and leaf nodes are tractable distributions, such as Bernoulli and Gaussian distributions [7, 10]. More precisely, letting the *scope* of an SPN be the set of variables appearing in it, a recursive definition is given as follows:

(1) a tractable univariate distribution is an SPN (the base case); (2) a product of SPNs with disjoint scopes is an SPN (denoting a factorisation over independent distributions); and (3) a weighted sum of SPNs with the same scope, where the weights are positive, is an SPN (denoting a mixture of distributions).

chinaXiv:202211.00326v1

ChinaXiv合作期刊

Formally, SPNs encode a function for every node that maps random variables $X_1, \ldots, X_n$ to a (numeric) output. For node j, the corresponding function f_j maps a world $X_1 = x_1, \ldots, X_n = x_n$ to its probability if it is a leaf node, the weighted sum $\sum_i w_i f_i^j(x_1,\ldots,x_n)$ for sum nodes where f_i^j are the children of node j, and $\prod_i f_i^j(x_1,\ldots,x_n)$ for product nodes. Conditional probabilities are expressed using:

$$\Pr(x_1,\ldots,x_m \mid x_{m+1},\ldots,x_n) = f_0(x_1,\ldots,x_n) / f_0(x_{m+1},\ldots,x_n) \tag{9}$$

where 0 is the root node, and the notation f_0 with only a subset of the arguments used in the denominator denotes marginalisation over the remaining arguments. The benefit of SPNs here is that a bottom-up pass allows us to compute conditional queries in time polynomial in the circuit size. (See, for example, [46] on other data structures that allow a more expressive class of queries.)

3.3 Notation

During our discussion, we use the following notation. Random variables are denoted using X or Y as well as X_i. The set of possible values for a variable (or feature) X_i are implicitly understood to be binned as intervals $\{x_{1i}, x_{2i}, \ldots, x_{ki}\}$. Abusing notation, we treat x_{ai}^1 and x_{ai}^0 to true and false values accorded to the Boolean variable serving as a propositional abstraction for the interval x_{ai}. We periodically generalize x_{ai} with x_i for a random variable X_i, where x_i represents any given interval $\leq k$. We denote a data set as D, with T being the set of instances, and V being the set of variables. D' represents a binarized data set, that is, after binning the points in D as intervals and interpreting those intervals as Boolean variables. We refer to queries as q and we also refer to log-likelihood function as $\mathcal{L}$. For WMISPNs and SPNs, we use f_j to refer to a general node j with f_0 referring to the root node and f_i^j referring to a node i with parent node j.

4. TRACTABLE MODEL STRUCTURE LEARNING

In this section we present our proposed algorithm which derives the structure and parameters for SPNs with piecewise polynomial leaves, equipped with WMI (WMISPNs). The method LearnWMISPN is capable of deriving such an SPN structure from hybrid data and further, it allows for the retention of the distribution parameters in the continuous case for use in advanced querying.

The structure learning approach for LearnWMISPN is derived from LearnSPN [10]. LearnSPN is a recursive top-down learning method which is capable of learning the structure and weights for SPNs by identifying mutually independent variables and clustering similar instances. LearnSPN takes a simple approach to structure learning by recursively splitting data into a product of SPNs over independent variables, or a sum of SPNs comprising subsets of instances. More generally, LearnSPN represents an algorithm schema which allows for a modular approach in structure learning from differing domains, which is why our approach for learning changes only the base case.

The primary extension with LearnWMISPN is the addition of generated polynomial leaf nodes. Given a data set with discrete, categorical and continuous attributes, LearnWMISPN learns polynomial weight functions as density approximations for the continuous cases, mapped to leaf nodes. By doing so,

chinaXiv:202211.00326v1

ChinaXiv合作期刊

LearnWMISPN provides a model which is capable of complex querying in the continuous case, while also conditioning over other variable types for hybrid domains.

Prior to structure learning, a preprocessing step is performed which first transforms discrete, categorical, and continuous features into a binary representation. Much like the algorithm schema of LearnSPN, the preprocessing step can implement any number of unsupervised binning methods including using a *mean split, equal frequency binning,* or *equal width binning* [47]. This is followed by a polynomial learning function which identifies the polynomial function coefficients and probability densities for the continuous features in the hybrid domain without prior knowledge of the true density function.

Algorithm 1 describes the recursive structure learning algorithm for LearnWMISPN and Figure 3 illustrates the recursive pipeline. LearnWMISPN takes as input the preprocessed binary data set D' with T instances and V' variables and returns a WMISPN representing the distributions of the hybrid domains. LearnWMISPN recurses on subsets of V' until a vector of unit length is found, at which point a corresponding univariate distribution is returned.

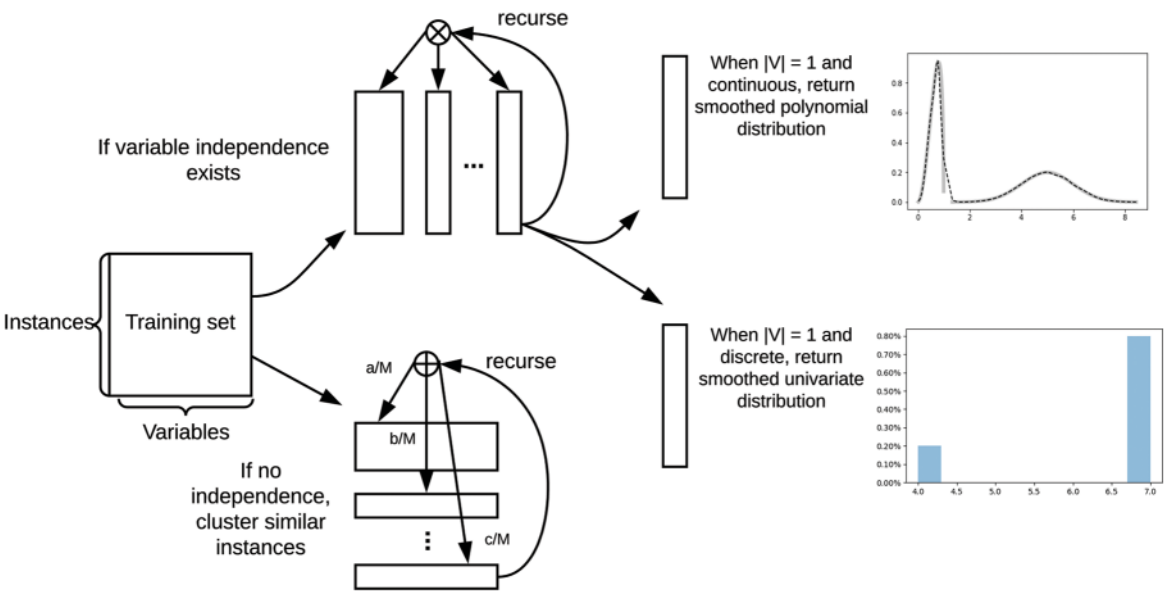


Figure 3. A recursive algorithm for learning hybrid domains.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

Algorithm 1. LearnWMISPN (T, V').

Input : T: set of instances, V': set of variables, θ: parameters (cf. Algorithm 2)
Output: SPN representing a distribution over V' learned from T

```
if |V'| = 1 then
    if variable V' is an instance of a continuous feature then
        return univariate distribution estimated
          from polynomial probability densities
    else
        return univariate distribution estimated
          from binary counts in T
    end
else
    partition V' into approximately independent subsets V'_j
    if success then
        return ∏_j LearnWMISPN(T, V'_j);
    else
        partition T into subsets of similar instances T_i
        return Σ_i (|T_i|/|T|) LearnWMISPN(T_i, V')
    end
end
```

The main novelty in LearnWMISPN is the handling of the base case. Without any prior knowledge, LearnWMISPN will return a smoothed univariate distribution for discrete and categorical variables, or piecewise polynomial distribution for continuous variables. These two outcomes set LearnWMISPN apart from LearnSPN, and this is represented with polynomial leaf nodes corresponding to continuous distributions.

If the base case has not been reached, then LearnWMISPN continues the recursive structure learning of LearnSPN. For the decomposition step, a variable split is identified which results in mutually independent subsets, generating a product node for the resulting subset. Mutual independence for the variable splits is determined by a G-test for pairwise independence:

$$G(x_1, x_2) = 2\sum_{i=0}^{1}\sum_{j=0}^{1} c(x_1^i, x_2^j)\log\frac{c(x_1^i, x_2^j)\,|T|}{c(x_1^i)c(x_2^j)} \tag{10}$$

where the summations range over the elements of each variable interval x_1 and x_2, $c(\cdot)$ refers to the occurrence count for a setting of a variable pair or singleton, and T refers to the number of instances [10].

As an example, say we are checking the independence for two continuous features $X_1 = x_1$ and $X_2 = x_2$ (we ignore binning to focus on the test itself). As there are only four possible combinations from the count function $c(x_1^i, x_2^j)$, we derive a 4×4 matrix which list the counts for $c(x_1^0, x_2^0)$, $c(x_1^0, x_2^1)$, $c(x_1^1, x_2^0)$, and $c(x_1^1, x_2^1)$. We also calculate a count vector for x_1 and x_2, respectively that counts the number activations (1) and negations (0) in each feature separately (represented by $c(x_1^i)$ and $c(x_2^j)$ in the equation). The empty sets Ø in the count vectors are then counted (it is only ever possible to have 1 empty set or 0 empty sets). From there we sum over the possible activation/negation cases using the G-test algorithm, and the final G-test value must be less than the threshold value calculated from features x_1 and x_2 to be classified as independent.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

The threshold value is calculated as $(2 \times DOF \times gfactor + 0.001)$. We can calculate the degrees of freedom (DOF) based on the number of states a feature can be represented by (2 in our case as it is either 0 or 1), and the empty set count for each feature $(2-\varnothing_{x_1}-1)\times(2-\varnothing_{x_2}-1)$. The *gfactor* is chosen as 0.001 and we add 0.001 to prevent a threshold of zero.

It is also noted that we perform the G-test on two features, say x_{ai} and x_{bi}, derived from a continuous feature X_i where a and b refer to a bin interval such that $1 \leq (a, b) \leq k$ given $\{x_{ai}, x_{bi}\} \subseteq X_i$. If x_{ai} has a large number of activations and x_{bi} has a small number of negations (mostly 0's with say a single activation 1), the G-test will still be able to determine that the features are dependent. We remark that the G-test is not a perfect metric for determining independence, but generally, we found it very successful when provided with binarized data.

For the conditioning step, similar instances are clustered using hard incremental expected-maximization (EM) generating a sum node. The ratio split on the clustered subset of instances determines the corresponding weight and what is returned is the sum of the resulting weighted clusters. We condition initially on a slice $S \subseteq D$, which is a data structure composed of a subset instances ($T_S \subseteq T$) and a subset of features ($V_S \subseteq V$) from the initial data set D. Clusters $\mathbb{C}$ are derived from slice S via 10 restarts through the subset of instances T_S four times in random order. A new cluster C is added to $\mathbb{C}$ if the log-likelihood is greater than the cluster penalty likelihood $-\sigma|V|+\sum_{v\in V_S}\ln\frac{1}{1+F_v}$, where σ is the cluster penalty and F refers to the attribute count for each feature (2 in our model). In order for a set of clusters to remain in the WMISPN structure and for learning to continue, the best set of clusters $\mathbb{C}$ is greedily chosen based on log-likelihood $\mathcal{L}$ improvements:

$$score = \sum_{t\in T_S} g(\mathcal{L}(C^- \mid S_t), C^{prior} + \mathcal{L}(C \mid S_t)) \;\forall\, C \in \mathbb{C} \tag{11}$$

where C^{prior} refers to the ratio of local instances T_C to slice instances T_S, S_t refers to the partition of data based on instance t, and C^- refers to the previous cluster in the recursive formula $g(\cdot,\cdot)$. The function $g(\cdot,\cdot)$ is used to calculate the maximum for the log-domain:

$$g(x,y) = \begin{cases} x + \ln(1+e^{(y-x)}) \text{ if } x > y \\ y + \ln(1+e^{(x-y)}) \text{ otherwise} \end{cases} \tag{12}$$

LearnSPN assumes a naive Bayes mixture model, where all variables are independent conditioned on the cluster; however LearnWMISPN applies a prior:

$$\Pr(V') = \sum_i \Pr(C_i)\prod_j \Pr(X_j \mid C_i)\mathbb{1}_c \Pr(X_j) \tag{13}$$

where C_i is the ith cluster and X_j is the jth variable, but the prior $\Pr(X_j)$ is only applied when the feature is continuous.

Overall, we upgrade both learning and querying over the models to handle piecewise polynomial density approximations. For the learning part, since only the base case is changed for our method, it suffices to show that the learning of the univariate distributions can be effective, which we discuss below. The querying is treated in the subsequent section.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

Formally, let $X_i = (x_{1i}, \ldots, x_{ki})$ where k is the number of intervals for every continuous feature X_i. The polynomial learner provides to LearnWMISPN $\lambda_{x_{ai}}$ a probability constant representing priors for the continuous interval x_{ai}, a being an arbitrary interval $\leq k$. In the base case of LearnWMISPN, a univariate distribution is returned but calculation is contingent on whether the feature is continuous or discrete. Given a continuous feature interval x_{ai}, we wish to find the calculation of the unnormalized probability of a WMISPN $f_l^p(x_{ai})$. Here p represents the parent node to the child leaf l node f_l^p which maps to an interval x_{ai}.

The leaf likelihood $f_l^p(x_{ai})$ for an interval x_{ai} is calculated based on activations and negations of instances for the interval. For activations, we find the count activtions $c_{ai}^1 = c(x_{ai}^1)$, and negations $c_{ai}^0 = c(x_{ai}^0)$. The likelihood for x_{ai} is then calculated as:

$$f_l^p(x_{ai}) = \frac{c_{ai}^1 \lambda_{x_{ai}}}{c_{ai}^1 \lambda_{x_{ai}} + c_{ai}^0 (1 - \lambda_{x_{ai}})} \tag{14}$$

This calculation considers the one-hot encoded representation of x_{ai} as negations correspond to the probability of $(x_{1i}, \ldots, x_{(a-1)i}, x_{(a+1)i}, \ldots, x_k)$ being activated instead.

Preprocessing Step: A preprocessing step is required to first transform the discrete, categorical, and continuous variables into a binary representation. The methods for converting a data set into a binary representation is contingent on the variable type in the hybrid data set. The preprocessing pipeline is summarized in Algorithm 2.

Algorithm 2. Preprocessing step(D, f, k).

Input : D: dataset of real values, **f**: vector of domain types, k: number of bins
Output: D': binary representation of D, $\mathcal{M}$: matrix of coefficients for the polynomials, **p**:vector of probability densities
$D' \leftarrow$EqualWidthBinning(D, **f**, k)
$\mathcal{M}$, **p** $\leftarrow$PolynomialLearner(D, **f**, k)

The first sub-task converts the real-valued data set D into a binary representation D'. In our model, equal-width binning was performed on continuous features, with the number of equal-width bins specified as a parameter. (A meta-learning step can be designed to choose the optimal number by studying log-likelihood or polynomial improvements.) Higher bin counts would increase the representative power and improve accuracy for complex queries on continuous distributions (see section 5). Each new bin generated from a feature represents a discrete range on that continuous distribution. A one-hot encoding transformation was performed on the expanded feature set to get a binary representation. For discrete and categorical variables, the binary representation is achieved again with a one-hot encoded transformation with equal width binning not taken into consideration.

As an example for one-hot encoding in the discrete case, consider a random variable Y which has a domain of boolean values. Booleans result in two classification labels so Y is expanded to (y_1, y_2), which now correspond to each boolean instance. Formally:

chinaXiv:202211.00326v1

ChinaXiv合作期刊

$$Y(x) = \begin{cases} y_1 = 1 \; if \; x \\ y_2 = 1 \; if \; \neg x \end{cases} \tag{15}$$

The two cases for Y are then represented as $[x] \rightarrow [1, 0]$ and $[\neg x] \rightarrow [0, 1]$, respectively. Now consider the continuous case, where equal width binning is implemented, we say that random variable X is defined as having a range $[a, b] \Rightarrow x \in \mathbb{R} : a \leq x \leq b$. If we split X into say three bins (x_1, x_2, x_3), with the split point defined as $s = (b - a)/3$ then we again have another representation for one hot encoding. Formally:

$$X(x) = \begin{cases} x_1 = 1 \text{ if } \; a \leq x < (a + s) \\ x_2 = 1 \text{ if } \; (a + s) \leq x < (a + 2s) \\ x_3 = 1 \text{ if } \; (a + 2s) \leq x \end{cases} \tag{16}$$

So for the minimum and maximum values in X we get the following one-hot encoded representation $[a] \rightarrow [1, 0, 0]$ and $[b] \rightarrow [0, 0, 1]$. This means the binary transformation of an instance $D_{i,:} = [b, x]$ is $D'_{i,:} = [0,0,1,1,0]$.

It is worth noting that increasing bin intervals leads to overfitting of the data and spurious likelihood values. Consider a data set D with M instances and a N-sized feature space, and consider the effect of increasing interval counts k (where $k > M$) on a given continuous random variable X_i where $\sum_{n=1}^{k} c(x_{ni}^1) \leq M$ and $\sum_{n=1}^{k} c(x_{ni}^0) \leq (k - M)$. Binning results in an increasing feature space N, which leads to continuous intervals x_i frequently being 0. This implies that prior likelihoods of activations decreases, while prior likelihoods for deactivations increases, which can also be seen from the one-hot encoded representation that leads to additional features with no values. Such problems are to be expected with overfitting; however, as we will see in the following section, we use an information criterion to regularize bin counts, choose an optimal number of bins and temper the piecewise polynomial function degree in the model. This is further evidenced by our empirical evaluations (*cf.* Table 2 and 3), which shows that our results naturally converge to ≤ 5 bins per continuous feature.

Polynomial Learner: The second task learns a piecewise polynomial (PP) approximation for any univariate probability density function (PDF) without prior knowledge or simplifying assumptions. The original data set D, the indices for the continuous variables, and the number of bins k are taken as input, and as output, we receive a vector of probability densities for each continuous feature bin. This bin information is retained in its respective polynomial leaf node in the SPN for use in complex querying.

The piecewise polynomial functions are calculated through a linear combination of b-splines as described in [48]. B-splines are polynomial curves that are right-side continuous, differentiable, and they form a basis in the space of piecewise polynomials. This follows that linear combinations of b-splines can represent any piecewise polynomial function[5].

[5] Note that we may place additional constraints on the learning regime. For example, [17] show that they can find a candidate hypothesis $f \in P_{k,n}(I)$ such that $||f - g||$, g being the unknown true density, can be made arbitrarily small. We chose the basis-spline technique, but none of our technical results hinge on opting for either of these methods.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

B-spline interpolation can then be used to find an approximation of the density function as a linear combination of ($Z = k + r - 1$) b-splines, where r defines the degree order spanning the whole domain. Thus, for learning a given b-spline, only the order-r which is equivalent to $r = degree - n + 1$, the number of intervals defined by a binning method-k, and the knot sequence ($\delta = (a_0, \ldots, a_k)$) which refers to where pieces of the polynomial intersect are required. We define the j^{th} b-spline $B_j^r(x)$, where $j = 1, \ldots, Z$ as follows:

$$B_j^r(x) = (a_j - a_{j-r})H(x - a_{j-r})\sum_{t=0}^{r}\frac{(a_{j-r+t} - x)^{r-1}H(a_{j-r+t} - x)}{w'_{j-r}(a_{j-r+t})} \tag{17}$$

where $w'_{j-r}(x)$ is the first derivative of $w_{j-r}(x) = \prod_{u=0}^{r}(x - a_{j-r+u})$ and $H(x)$ is the Heaviside function:

$$H(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{18}$$

Our objective now is to learn piecewise polynomial (PP) weights for each of the intervals. So suppose $g : I \to \mathbb{R}_+$ is the density of an unknown distribution over I, where I is an interval. We would like to find a candidate PP hypothesis f. We say f is a k-piecewise, degree-n polynomial(also called mixture of polynomials (MOP)) if there is a partition of I into k disjoint intervals $(I_1, \ldots, I_k)$ such that $f(x) = f_j(x) \forall x \in I_j$, and each $(f_1, \ldots, f_k)$ is a polynomial of degree $\leq n$. Let $P_{k,n}(I)$ be the class of k-piecewise degree-n polynomials over I. We would now like to draw $m \leq |D|$ samples that finds m candidate hypothesis $f_1, \ldots, f_m \in P_{k,n}(I)$, and outputs $f^* \in \langle f_1, \ldots, f_m \rangle$ that maximizes the likelihood of observing the samples. Note that k is fixed by the intervalisation step, and so we only need to iterate over the maximum degree of the polynomial and that we are only learning univariate distributions for the leaf node.

The order, and, if not previously specified, the number of bins are chosen during training time by means of the Bayesian Information Criterion (BIC) [49].

$$BIC(x, f(x)) = \mathcal{L}(x \mid f(x)) - \frac{log(|D|)}{2} \tag{19}$$

Where $\mathcal{L}$ denotes the log-likelihood. Use of BIC scores and b-spline interpolation follows the usual properties of probability density functions, such as guaranteeing that MOP approximations remain continuous, integrate to one, are non-negative, and free of prior assumptions [50].

The BIC scoring function has been shown to be robust and was chosen to avoid overfitting the model parameters. In addition, the chosen method favours smaller polynomial ranks [50] which improves the efficiency of the integration without losing accuracy. We reiterate that we do not make any assumptions about the true density[6].

[6] It is interesting to observe that B-splines have been influential in a number of disciplines, including physics, engineering, and computer vision, and our algorithm contributes, for the first time, "deep B-splines". For the future, it would be interesting to study how LearnWMISPN would be applied in these domains, for modelling latent dependencies in computer graphics, for example.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

5. PROBABILISTIC MODEL QUERYING

This section is devoted to expanding out definition of querying with respect to WMISPNs. We first detail the means by which SPNs perform tractable inference, then we illustrate our integration of complex querying into the WMISPN framework.

SPNs provide tractable querying on univariate distributions and can calculate the marginal and conditional probabilities on learned features. Other schemas for SPNs limit the scope of querying to binary activations $\Pr(X_i = 1)$ to the leaf nodes [7]. In WMISPNs, we are able to expand our queries to complex cases where leaf nodes maps X_i to new probability ranges $(a \leq x \leq b)$, $a, b \in \mathbb{R}$.

5.1 SPN Inference

General SPN inference is initiated at the leaf nodes where queries are presented in the form of selected activations of the random variables $q(X_1 = 1, X_2 = 0, \ldots, X_N = 1)$. The mapped probabilities x_i propagate upward from the leaf nodes where values from children nodes are either multiplied at product nodes, or the weighted sum is taken at sum nodes. The final likelihood of the queries is returned at the root node 0.

To normalize the returned likelihood, a partition function is required to calculate the normalization constant. Formally:

$$Z = \sum_i f_0(x_i) \tag{20}$$

Inference is deemed tractable due to the partition function being computational in time linear to the number of edges in the learned SPN, which in turn results in queries that can be calculated in time linear $\Pr(q) = f_0(x_1, 1 - x_2, \ldots, x_n)/Z$. The same is true for conditional likelihood estimates.

5.2 Complex Querying

We extend the SPN inference model to allow complex queries where inference can be performed over continuous as well as discrete features, and combinations thereof: for example, queries on discrete features such as $male(X)$ conditioned on continuous ranges such as $40 \leq weight(X) \leq 50$. The advantage of complex queries, with their capability of performing inference on conjunctions of discrete and continuous feature ranges, allows modellers added ability in assessing the relationship of probability distributions in data. Taking our example of credit risk, complex querying can provide means to model the decision boundary, provided a trained model and a mixed discrete continuous data set. An exhaustive querying of a model with complex queries can determine what feature values would lead to a classification change. Taking the credit risk example again, complex queries of the form $\Pr(\texttt{class} = BAD | 0 < \texttt{creditamount} < k)$ and $\Pr(\texttt{class} = GOOD | k \leq \texttt{creditamount} < 10{,}000)$ where k indicates a real value boundary for the feature creditamount, can inform the modeller where the likelihood of classification changes. In this example we focus on one feature but complex queries allow for multiple conjunctions as will be explained further.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

chinaXiv:202211.00326v1

First, observe that LearnWMISPN is given probability constants as well as polynomial representations for continuous intervals based on the BIC score, thus standard univariate boolean queries do not require integration of new probability constants. In particular, suppose $L(S\ PN)$ refers to the propositional abstraction of the SPN (that is, think of intervals as propositions and only refer to the probability constants), and let $PP(S\ PN)$ refer to the continuous representation. We can then describe a query $q \in L(S\ PN)$ to mean a Boolean query defined as $q(x_1 = b_1, \overline{x_1} = 1 - b_1, \ldots, x_n = b_n, \overline{x_n} = 1 - b_n)$ where $b_i \in \{0, 1\}$.

Proposition 1 *Suppose T is a learned WMISPN. Then the partition function of T for any $q \in L(T)$, the probability of q can be computed in time linear to the number of edges in T.*

Our system augments the above-described machinery for querying to allow for complex queries. This adds the ability to ask queries over arbitrary continuous ranges which in turn enables a mixed querying setup between continuous and discrete features in the same formulas.

Standard SPN inference is initiated at the leaf nodes where queries are presented in the form of selected activations of the random variables (i.e., a state). The mapped probabilities propagate upward from the leaf nodes where values from children nodes are either multiplied at product nodes, or the weighted sum is taken at sum nodes. The final likelihood of the queries is returned at the root node.

Inference for continuous attributes is based on WMI. Let us explain the intuition using a univariate distribution. Our structure learning algorithm above assigns ranges $r_i = [\alpha_i, \beta_i]$ and their piecewise polynomial density approximation $p_i(x)$ to a continuous SPN leaf node i. With that, we can perform inference at the leaf node itself through WMI. For example, assume a continuous leaf node for the attribute "weight" has the range $34 \leq weight(X) \leq 55$ and suppose its calculated polynomial is $-0.051+0.0016x$. The probability for the query $40 \leq weight(X) \leq 50$ can then be calculated using $\int_{40}^{50} -0.051 + 0.0016x\ dx = 0.21$. Once the leaf has processed the query its value is passed to the SPN where it is applied to the other variables of the query. We reiterate that inference using the polynomials is not performed in the SPN itself. Only the value is passed on to be interpreted. This demonstrates the built-in modularity of our approach.

Naturally, posing a query such as $(40 \leq weight(X) \leq 60)$, spanning over multiple intervals is less trivial. In this case, assume a second interval for $weight(X)$ is specified $55 \leq weight(X) \leq 77$ with a polynomial density of $-0.1469-0.0019x$. Then, the probability of the query is calculated as $\Pr(40 \leq weight(X) \leq 70) = \Pr(40 \leq weight(X) \leq 55) + \Pr(55 \leq weight(X) \leq 70) = \int_{40}^{55} -0.051 + 0.0016x\ dx + \int_{55}^{60} 0.1469 - 0.0019x\ dx = 0.375 + 0.291 = 0.666$. In general, as the ranges were defined to be mutually exclusive, the calculation of the probability boils down to:

1). dividing the range into m–k subranges where each corresponds to a leaf node i, $i \in \{1, \ldots, n\}$, $1 \leq k \leq m \leq n$;
2). performing inference through WMI in each leaf with the polynomial density approximation $p_i(x)$; and then
3). adding the calculated probability values to obtain the final value.

ChinaXiv合作期刊

That is,

$$\int_a^b p(x)dx = \int_a^{\beta_k} p_k(x)dx + \sum_{j=k+1}^{m-1} \int_{\alpha_j}^{\beta_j} p_j(x) + \int_{\alpha_m}^{b} p_m(x)dx \tag{21}$$

By extension, if a query consists of multiple continuous features $X_1, \ldots, X_N$, we find that we require a joint density approximation over the variables in order to infer the likelihood. We recall that given multiple dependent variables, the joint likelihood is calculated via $\Pr(X_1, \ldots, X_N) = \Pr(X_1, \ldots, X_{N-1}) \Pr(X_N|X_1, \ldots, X_{N-1})$ as per the chain rule. We also recall Equation (9) for the SPN conditional likelihood and apply it WMISPNs where we see the likelihood over dependent features presented as:

$$\Pr(X_1, \cdots, X_N) = \frac{f_0(X_1)}{Z} \frac{f_0(X_2, X_1)}{f_0(X_1)} \cdots \frac{f_0(X_N, X_{N-1}, \cdots, X_1)}{f_0(X_{N-1}, \cdots, X_1)} \tag{22}$$

This can be reduced to $f_0(X_1, \ldots, X_N)/Z$. Effectively, we are able to represent the joint density function $P^{1\ldots N}(X_1, \ldots, X_N)$ with a WMISPN model over the variables, and perform inference on the continuous features using the leaf nodes to integrate over the ranges:

$$\int_{a_1}^{b_1} \ldots \int_{a_N}^{b_N} p^{1\cdots N}(X_1, \cdots, X_N) dx_1 \cdots dx_N \tag{23}$$

When integrating the model into the SPN some SPN properties need to be considered. The root node will always calculate the likelihood of a query by taking the product of its children. Given the mutual exclusivity of a continuous feature's multiple ranges, two or more passes are required for an SPN to calculate the likelihood of a query atom over multiple intervals for the same feature.

For any univariate query $q(x)$ of the form $(a \leq x \leq b)$, we define the query length to be the number of propositions $(x_1, \ldots, x_k)$ such that $q(x) \cap x_i^+ \neq \{\}$. Here x_i^+ is the refinement of the boolean activation which retrieves the interval it corresponds to. As an example, suppose leaves in the WMISPN mention $\alpha \leq x \leq \beta$ and $\beta \leq x \leq \gamma$. Then a query of length 1 is of the form $\alpha^* \leq x \leq \beta^*$, where $\alpha \leq \alpha^*$ and $\beta^* \leq \beta$. A query of length 2 is of the form $\alpha^* \leq x \leq \gamma^*$, where $\alpha \leq \alpha^*$ as before but $\beta < \gamma^* \leq \gamma$. In essence, complex querying finds the number of intervals in the WMISPN that $q(x)$ intersects with.

Proposition 2 *Suppose T is a learned WMISPN and $q \in PP(T)$. The probability of the univariate interval query q of length k requires k passes through the WMISPN.*

If $q \in L(S\ PN)$ then query length is 1, we then immediately get Proposition 1 as a special case. For simplicity, we assume that queries are limited to the following syntax:

1). Every query atom is either an interval $x \in [a, b]$ or a discrete feature $A \in [a_1, \ldots, a_n]$.
2). A query consists of a number of conjunctions.
3). Each conjunction has to be distinct, e.g. no conjunction shares the same query atoms.

Here, (2) and (3) are not fundamental restrictions: it is easy to extend (2) by applying the rule $\Pr(A \vee B) = \Pr(A) + \Pr(B) - \Pr(A \wedge B)$, and in the case of (3), linear arithmetic solvers can be used to reason about multiple constraints for the same variable: for example, $\Pr(30 \leq weight(X)) \wedge \Pr(50 \leq weight(X) < 60)$ can

chinaXiv:202211.00326v1

be resolved to $\Pr(30 \leq weight(X) < 60)$. So, (1) says computing the probability of expressions such as $x > y$ and $x + y > 2z$, where x, y, z are continuous variables, cardinality queries [8], among others are not dealt with currently.

We remark that in [30], the tractable inference advantages of MSPNs were utilized for methods akin to our complex querying. The proposed Model-based Approximate Query Processing (AQP) takes advantage of MSPN inference, which while similar to our method, is specifically concerned with SQL style queries and speed of inference. As the queries posed are SQL in nature (user-defined functions, arithmetic expressions, etc.) the scope of inference queries is larger than that found with complex queries as defined here. [30] utilizes both probabilistic and sampling approaches for inference, specifically with sampling serving to aid in approximating complex aggregation queries. Our account is focused on exact inference currently.

There are some shared techniques, such as readjusting of leaf weights for inference. [30] uses a sampling method with conditioned constraints, while our method uses the interval range on a given continuous feature to determine the weight adjustment at the leaf node. As mentioned, our approach is exact, does not use sampling methods, and as we are not investigating increasing inference speeds we find pruning of our WMISPN model unnecessary whereas Kulessa et al. actively do this. As [30] uses MSPNs, they calculate continuous variables with linear interpolations of histograms, as such their inequality queries are limited to linear approximations on continuous intervals (see Section 2) where they are treated as constants preventing user-specified inequality type queries (see Equation (6)). With our framework, we find our piecewise polynomial approximations allow for user-defined complex querying. On the other hand, the SQL mechanism has advantages such as biased sampling on rare sub-populations for ad-hoc querying as well as providing general SQL aggregation queries such as count, average, and sum. As such, we find it a useful research direction to combine complex querying with the SQL style queries with aggregation.

chinaXiv:202211.00326v1

6. EXPERIMENTAL EVALUATION

The goal of this section is to evaluate the merits of LearnWMISPN and the complex query interface. Specifically, we attempt to address the following questions:

- **Q1** How effective (in terms of log-likelihood) is LearnWMISPN on complex mixed discrete-continuous data?
- **Q2** How reasonable is the learned distribution: that is, what is the order of the learned polynomials, and what is the spread of the probabilities for the underlying intervals?
- **Q3** How effective is the query interface in the presence of increasing query lengths?

We measured the performance of the learned SPNs on preprocessed hybrid domain data sets discussed in the independent effort [28] that introduced Mixed-Sum Product Networks (MSPNs), Automatic Bayesian Density Analysis (ABDA) [31], and Bayesian SPNs (BSPNs) [37], as discussed in Section 2.

chinaXiv:202211.00326v1

Our performance evaluation of SPNs in conjunction with WMI was performed using three different regimes. We measured the performance of the learned SPNs from LearnWMISPN on preprocessed hybrid domain data sets. Doing so allowed us to directly compare the performance of LearnWMISPN to learn [28, 31, 37], and test whether our approach with continuous distributions was effective. We then investigated whether further increasing the number of leaves per continuous feature resulted in improvements in the model. Next, measuring the complex querying capacity of WMI with SPNs was performed by recording the computation times dependent on query length. We then demonstrate complex querying accuracy by measuring error rates using synthetic data. As we mentioned before, WMI is a very flexible framework for inference in hybrid domains, and as our empirical results show, our integration with SPNs has yielded a *scalable* unsupervised learning architecture that is able to handle benchmarks and many (preprocessed) UCI data sets involving hundreds of variables.

Q1 Hybrid Benchmarks: We evaluated LearnWMISPN on 11 hybrid domain data sets taken from the UCI machine learning repository [51]. Each data set was composed of differing proportions of categorical, discrete, and continuous features. The diverse set of domains comprised data from financial, medical, automotive and other sectors. The hybrid data sets were selected based on previous research in learning on hybrid domains [28, 31, 37]. As we wanted the capacity to make direct comparisons to previous research, we primarily focused on the data sets used in other works. The most important factor was that the data sets remain mixed with discrete and continuous features to take advantage of our methods integration of piecewise polynomial approximations. MSPNs, ABDA, and BSPNs were used as a comparative baselines in our evaluation. MSPNs proved to be a successful mixed probabilistic model [28], and given the similar nature of our objectives, a comparison was appropriate. ABDA and BSPNs relied on MSPNs as a comparative baseline in their respective works, thus they both were included as comparative models as well. Structure learning with LearnMSPN was performed with different variants. MSPNs were trained using the Gower distance, which is a metric over hybrid domains, with Gaussian distributional assumptions for continuous variables. MSPNs were also trained using the randomized dependency coefficient (RDC) which does not make parametric assumptions. For each model, histogram representations were compared against isometric regression models.

The dimensions of the data sets can be seen in Table 1. Given the original data sets used real values, our preprocessing results in an augmented binary matrix that has a larger variable count. In the original work, more data sets were used to measure performance, but in our case, we omitted data sets which did not contain continuous features. We used the original LearnSPN training, validation, and test ratio splits, which were 75%, 10%, and 15%, respectively.

ChinaXiv合作期刊

chinaXiv:202211.00326v1

Table 1. Data set statistics.

Data set	$\|V\|$	$\|V'\|$	Train	Valid	Test
anneal-U	38	95	673	90	134
australian	15	50	517	69	103
auto	26	85	119	16	23
car	9	50	294	39	58
cleave	14	35	222	29	44
crx	15	54	488	65	97
diabetes	9	33	576	76	115
german	21	76	750	99	150
german-org	25	70	750	99	150
heart	14	35	202	27	40
iris	5	11	112	15	22

The log-likelihood results for Learn methods and LearnWMISPN are shown in Table 2. As can be seen, our model outperforms other models by a wide margin. The performance of our model results in significantly better likelihoods across the majority of data sets. In most cases, our implementation only required two bins per continuous feature, and one instance (the iris data set) required a three bin split in order to produce a likelihood competitive to the models.

Table 2. Average test set log likelihoods for structured learning methods on hybrid data sets.

Data set	WMI-SPN	Gower-		RDC-		ABDA	BSPN
	LearnWMISPN (bin size)	hist	iso	hist	iso	-	-
anneal-U	-14.54 (2)	-63.55	-38.83	-60.31	-38.31	**-2.65**	-16.44
australian	**-10.47 (2)**	-18.51	-30.37	17.89	-31.02	-17.70	-21.51
auto	**-27.12 (2)**	-72.99	-69.40	-73.37	-70.06	-70.62	-58.45
car	**-9.11 (2)**	-30.46	-31.08	-29.13	-30.51	-35.04	-28.73
cleave	**-13.82 (2)**	-26.13	-25.86	-29.13	-25.44	-22.60	-22.95
crx	**-10.52 (2)**	-22.42	-31.62	-24.03	-31.72	-15.53	-11.54
diabetes	**-6.29 (2)**	-15.28	-26.96	-15.93	-27.24	-17.48	-21.21
german	**-20.42 (2)**	-40.82	-26.85	-38.82	-32.36	-32.10	-26.76
german-org	**-21.14 (2)**	-43.61	-26.85	-37.45	-27.29	-26.29	-21.32
heart	**-14.87 (2)**	-20.69	-26.99	-20.37	-25.90	-23.39	-22.93
iris	-3.56 (3)	-3.61	-2.89	-3.44	**-2.84**	-2.96	-3.54

Note: The bin size is given for the WMISPN log likelihoods.

The effectiveness of the framework can be attributed to a number of factors. We observed that unsupervised binning interfaced with LearnSPN's existing structure learning performed well on the hybrid data sets. This can be thought of as piecewise constant WMISPNs. The approach is simple and thus, fast, adding negligible complexity to the original LearnSPN machinery: the effectiveness in learning binary representations and the simple counting of Boolean variable activations is inherited from LearnSPN. In addition to that, we support the learning of polynomial weights, and not just constant or linear ones. In particular, our use of the BIC criteria to choose the most optimal representation, which has a bias towards smaller bin numbers and polynomial exponents.

Q2 Model Complexity: When investigating the correlation of model accuracy to model complexity, we used data sets with the majority variables in the continuous domain. We also used data sets with significantly more instances to learn on, ranges being from 1,024 to 17,389 in order to demonstrate the scalability of WMISPNs. The data sets investigated were the Cloud data set, Statlog (Shuttle) data set, and a subset of the MiniBooNE particle identification data set, which were all from the UCI machine learning repository [51]. Our inquires below use equal width binning as a discretisation method for continuous features.

The most immediate question is what is the nature of learned polynomials. The BIC measure, as we mentioned, is used to determine the number of bins and polynomial order. So, by relaxing the criteria for the number of bins to be used, we can study the polynomials. In Table 3, we see statistics on the order of the learned polynomials for 2 *vs* 5 bins. The order never goes beyond 6, and in a majority of the cases is ≤4, confirming once again that the learned orders stay manageable. Increasing the bin size favours low-order polynomials: only *Australia* and *Cloud* have order 6 polynomials with 5 bins.

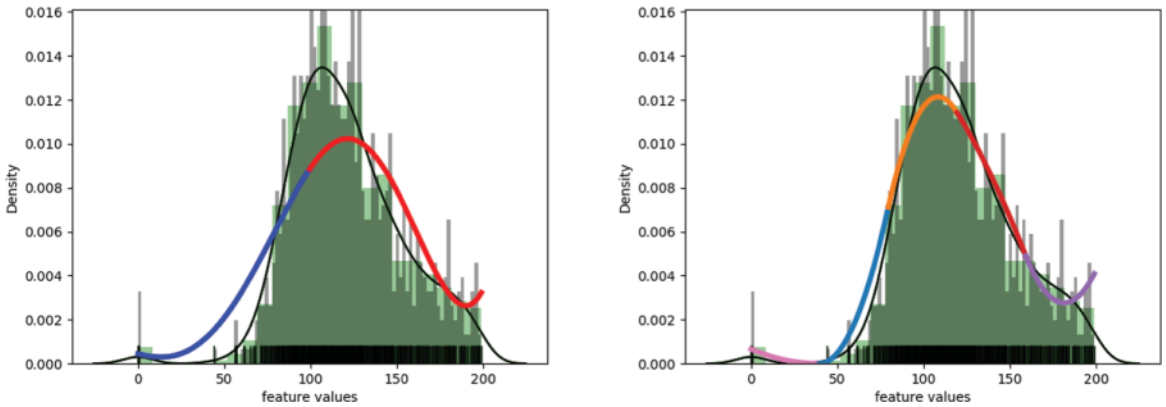
Table 3. Comparison of the distributions of orders (in %) for 2 and 5 bins.

Data set	Bins	2nd-Order	3rd-Order	4th-Order	5th-Order	6th-Order
Australia	2	0	16.667	16.667	33.3	33.3
Australia	5	16.667	33.333	16.667	16.667	16.667
Auto	2	15.8	36.842	31.579	12.789	0
Auto	5	73.684	15.789	10.526	0	0
German-org	2	0	33.333	0	0	66.666
German-org	5	33.333	33.333	0	33.333	0
Heart	2	0	20	20	60	0
Heart	5	0	60	40	0	0
Iris	2	50	0	50	0	0
Iris	5	75	25	0	0	0
Statlog	2	42.857	0	0	0	57.143
Statlog	5	28.571	57.143	0	14.289	0
Cloud	2	10	10	40	10	30
Cloud	5	30	50	10	0	10

In Figure 4, It is evident that increased binning results in piecewise polynomial functions that can better approximate the spread of data. In the case of the diabetes data set, at 5 bins the learned polynomial function was better able to capture the distribution of continuous feature points compared with the 2 bin approximation. The improved polynomial function approximation also lends itself to more accurate bin probabilities for SPN inference calculations. Also of note is the polynomial order for the diabetes continuous feature (Plasma glucose concentration) at 2 bins was a 4^{th} order polynomial, while at 5 bins the polynomial order was only 2.

chinaXiv:202211.00326v1

ChinaXiv合作期刊



(a) Diabetes 2 bin polynomial approximation (b) Diabetes 5 bin polynomial approximation

Figure 4. Comparison of learned piecewise polynomial functions for feature 2 (Plasma glucose concentration) of the diabetes data set. Note: Bin intervals are represented by alternating colors.

The second question, then, is how are the probabilities spread from the learned representation? (That is, assume we learn $p_1(x)$ for $0 \leq x \leq 5$ and $p_2(x)$ for $5 \leq x \leq 10$; we consider the probabilities on computing the volumes and normalising.) Naturally, this spread depends very much on the data, but by considering multiple data sets, one can empirically study the effectiveness of the learning regime. We see in Figure 6 that the representations match the characteristics of the data (e.g., sparseness for some attributes, missing values), diverse as they are.

The third natural question is whether learning polynomials are beneficial at all? We mentioned earlier that unsupervised binning along with a simple binarisation scheme can be seen as a simplistic hybrid model – an instance of piecewise constant WMISPNs – for which LearnSPN suffices. Clearly, such an endeavour would come at a significant loss of expressiveness, e.g., no interval querying. So, by letting BIC determine the polynomial order but explicitly setting the bin parameter, we can contrast LearnSPN and LearnWMISPN. It should be noted that as each bin range corresponds to a distribution represented by a given polynomial, further bin increases on a feature result in differing polynomial representations. (Analogously, classical SPNs will redistribute the spread of discrete probabilities.) In Figure 5, the plotted performance of average log-likelihoods over bin complexity is normalized by the number of new variables generated from equal width binning. We see that accuracy on the data set does increase as the number of bins per feature increases, and thus LearnWMISPN yields a more accurate representation.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

chinaXiv:202211.00326v1

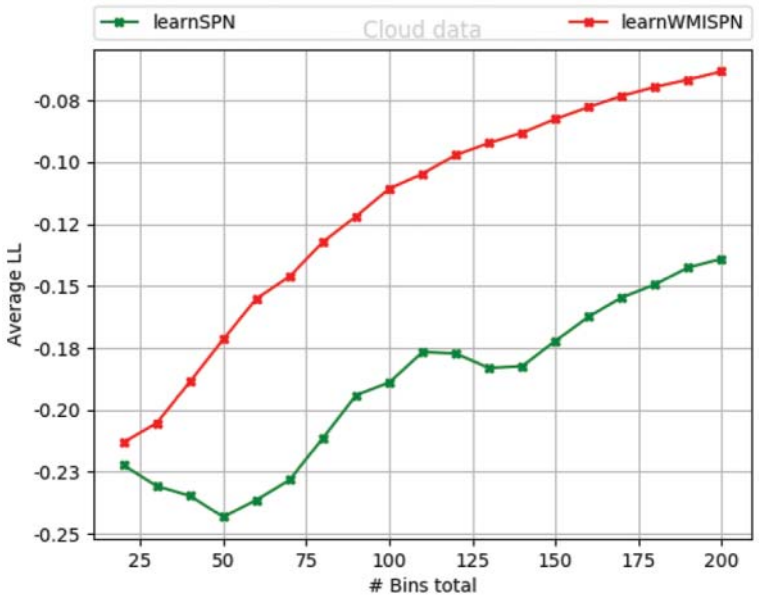


Figure 5. Instances of expanded Cloud data sets and resulting normalized log-likelihoods.

Comparisons done with LearnWMISPN (red) and LearnSPN (green) [10]. It should be noted that LearnWMISPN is yielding a highly granular and intricate representation. For example, at 5 bins per feature, the first bin of the first feature (the pixel mean) defined over a range of $(3.0 \leq x \leq 20.1)$ is given a density approximation that is a 3rd degree polynomial, and then at 15 bins per feature, over a range of $(3.0 \leq x \leq 8.7)$ the density approximation is now a 4th degree polynomial.

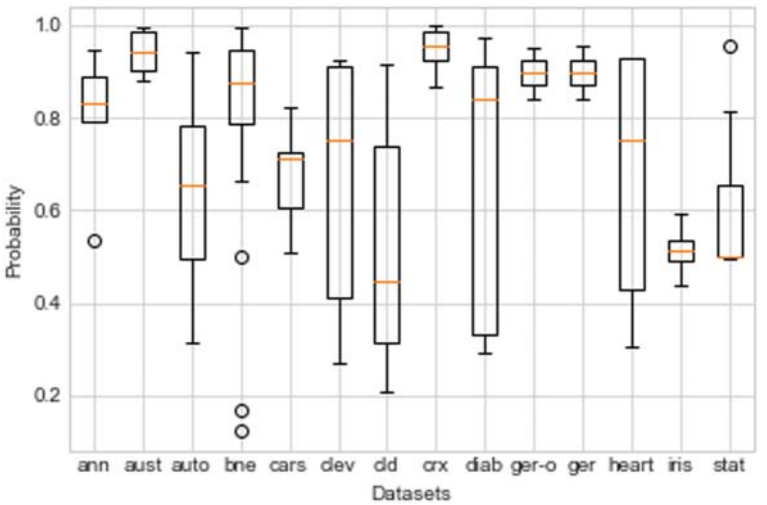


Figure 6. The spread of probabilities, with 2 bins per attribute.

ChinaXiv合作期刊

Q3 Query Interface: The capacity to query SPNs trained on continuous domains represents a significant improvement in terms of interpreting data. With regard to posing complex queries to WMISPNs, we studied the computation time for inferences. The three data sets used for measuring this are the Cloud, Statlog (Shuttle), and MiniBooNE data sets. All data sets are comprised of continuous features, with Statlog containing a single discrete feature. In order to measure complexity, we refer to query length (see Section 5) and refer to continuous queries as c^i and discrete as d^i with i representing the query count. We generated 10 random queries based on the continuous ranges for each data set, including discrete outcomes for Statlog, and averaged the inference time. We also fixed the number of bins per continuous feature to two.

In Table 4, we see the inference speeds on the complex queries. Overall, it is clear that complex queries do not slow down the model. For all query lengths, the time per query remains in the nanosecond range. As query length increases, more time is required but overall the increase in query time is linear. The model was also capable of handling queries with mixed continuous and discrete atoms, further demonstrating the capability of WMISPNs.

Table 4. Average query time for differing query lengths.

Data set	q_1	q_2		q_3		q_4		q_5	
	c^1	c^2	c^1, d^1	c^3	c^2, d^1	c^4	c^3, d^1	c^5	c^4, d^1
Cloud	1401	2154	n/a	2563	n/a	3659	n/a	4025	n/a
Statlog	1299	n/a	1878	n/a	2365	n/a	2749	n/a	2982
MiniBooNE	1168	2071	n/a	2219	n/a	2290	n/a	3048	n/a

Note: Time per query is measured in (ns/query).

Finally, in order to measure the accuracy of the learned models, we calculated the inference likelihood mean square error (MSE) for three synthetic data sets. The data sets comprised a 1000-instance sampled from a Gaussian $\mathcal{N}(0.7, 02)$, an exponential ($\lambda = 1$), and a distribution mixture $\mathcal{N}(0.5, 0.1)$ and Beta $\mathcal{B}e(2, 20)$. 150 complex queries were generated each of length 1. From our results in Figure 7, we observe the overall accuracy of the model in handling complex queries. The MSE also tends to approach lower values as the bin count increases, with the exception of the exponential distribution where the MSE was lowest with 3 bins but generally the rate remained steady. The learned WMISPN model also demonstrates robustness to handling inference on mixtures of distributions, again, seen by the decreasing error rate. Effectively our model is capable of performing both tractable inference, and returning accurate likelihoods for complex queries on various distributions.

chinaXiv:202211.00326v1

ChinaXiv合作期刊

chinaXiv:202211.00326v1

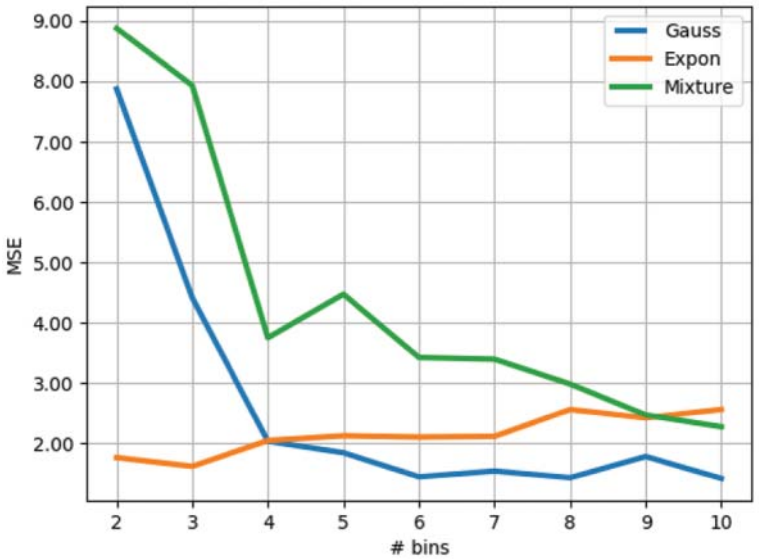


Figure 7. The inference likelihood mean square error for sampled data taken from Gaussian, Exponential, and mixture distributions of Gaussian and Beta.

It is worth mentioning again the comparisons here to MSPNs [28] and other works [30,31]. Regarding MSPNs, the differences in the general framework include our use of the G-test versus the use of the Renyi decomposition. As we abstract continuous features into discretized bins, it is sufficient to rely on the G-test. We also reiterate our use of piecewise polynomials over piecewise linear constants which allows our model to perform comparisons. We also point out inference is mixed concerning MSPNs, while our model is WMI based, and by doing so we can perform complex querying with respect to arbitrary intervals. Contrast that with the MSPN SQL approach [30] which uses the piecewise linear constants for inference on continuous random variables and is generally dissimilar to our methods (see Section 5) as SQL queries are more expansive. [31] also explores inference with MSPNs and replaces the piecewise linear approximations mapped to leaf nodes with a likelihood dictionary where parameters are learned via Gibbs sampling. Again, our model is agnostic to the distribution of the data and learns piecewise polynomial approximation which augments our inference mechanism with complex querying. Future research would benefit from the integration of all these methods, and we also find we are in a good position to go beyond intervals and consider oblique supports (see Section 7).

7. CONCLUSION

Deep architectures are powerful learning paradigms that capture the latent structure and have proven to be very successful in machine learning. Guessing the right architecture for complex data is challenging, and so paradigms such as SPNs are attractive alternatives in providing a *unsupervised* learning regime in addition to robust inference computations.

ChinaXiv合作期刊

Here, we pushed the envelope further to consider a systematic integration of SPNs and WMI, allowing us to learn tractable, non-parametric distributions and convex combinations thereof for hybrid data, also in a *unsupervised* fashion. The integration was achieved by minimally adapting the base case for an SPN structure learning module, which makes our approach generic to a large extent. Different from our predecessors, we show for the first time how tractable distributions of arbitrary granularity can be learned, and more importantly, how to query these distributions over a rich interval syntax. Our empirical results show that our implemented system is effective, scalable and incurs a very little cost for handling continuous features, all of which are very desirable for learning from big uncertain data. One interesting direction is to extend the underlying constraint language to handle oblique supports (e.g. $x > y$ and $x + y > 2$) rather than only intervals. A second challenge for the future includes extending the query language with features like counting operators, which would allow us to reason about the cardinality of sets of objects in an image, thus enabling an interface for commonsensical reasoning within deep architectures.

chinaXiv:202211.00326v1

ACKNOWLEDGEMENTS

Andreas Bueff was partly supported by EPSRC Platform Grant EP/N014758/1. Vaishak Belle was partly supported by the EPSRC grant *Towards Explainable and Robust Statistical AI: A Symbolic Approach*, and partly supported by a Royal Society University Research Fellowship.

AUTHOR CONTRIBUTIONS

A. Bueff (andreas.bueff@ed.ac.uk), S. Speichert (s.speichert@ed.ac.uk) and V. Belle (vaishak@ed.ac.uk) were all responsible for the design of the research, the implementation of the approach, the analysis of the results. All authors contributed to the manuscript writing and they edited and reviewed the final version of the article.

REFERENCES

[1] Murphy, K.: Machine learning: A probabilistic perspective. The MIT Press, Cambridge (2012)
[2] Bacchus F., Dalmao, S., Pitassi, T.: Solving #SAT and Bayesian inference with backtracking search. Journal of Artificial Intelligence Research 34(1), 391–442 (2009)
[3] Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM Journal on Computing 8(3), 410–421 (1979)
[4] Koller D., Friedman, N.: Probabilistic graphical models: Principles and techniques. The MIT Press, Cambridge (2009)
[5] Bach, F.R., Jordan, M.I.: Thin junction trees. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, pp. 569–576 (2001)
[6] Chavira, M., Darwiche A.: On probabilistic inference by weighted model counting. Artificial Intelligence 172(6–7), 772–799 (2008)
[7] Poon, H., Domingos, P.: Sum-product networks: A new deep architecture. In: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, pp. 337–346 (2011)

chinaXiv:202211.00326v1

[8] Rashwan, A., Poupart, P., Chen Z.: Discriminative training of sum-product networks by extended baum-welch. In: Proceedings of the Ninth International Conference on Probabilistic Graphical Models (PMLR), pp. 356–367 (2018)
[9] Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2(1), 1–127 (2009)
[10] Gens, R., Domingos, P.: Learning the structure of sum-product networks. In: International Conference on Machine Learning, pp. 873–880 (2013)
[11] Hsu, W., Kalra, A., Poupart, P.: Online structure learning for sum-product networks with Gaussian leaves. arXiv preprint arXiv:1701.05265 (2017)
[12] Liang, Y., Bekker, J., van den Broeck, G.: Learning the structure of probabilistic sentential decision diagrams. In: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI), pp. 1–10 (2017)
[13] Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20, 197–243 (1995)
[14] Yang, X., Cao, J., Yu, W.: Exponential synchronization of memristive cohen–grossberg neural networks with mixed delays. Cognitive Neurodynamics 8(3), 239–249 (2014)
[15] Fierens, D., et al.: Inference in probabilistic logic programs using weighted CNF's. In: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, pp. 211–220 (2011)
[16] Belle, V., Passerini, A., van den Broeck, G.: Probabilistic inference in hybrid domains by weighted model integration. In: Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI), pp. 2770–2776 (2015)
[17] Acharya, J., et al.: Sample-optimal density estimation in nearly-linear time. arXiv preprint arXiv: 1506.00671 (2015)
[18] Albarghouthi, A., et al.: Quantifying program bias. arXiv preprint arXiv:1702.05437 (2017)
[19] Sanner, S., Abbasnejad, E.: Symbolic variable elimination for discrete and continuous graphical models. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, pp. 1954–1960 (2012)
[20] Shenoy, P.P., West, J.C.: Extended shenoy–shafer architecture for inference in hybrid Bayesian networks with deterministic conditionals. International Journal of Approximate Reasoning 52(6), 805–818 (2011)
[21] Chan, S.-O., et al.: Efficient density estimation via piecewise polynomial approximation. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 604–613 (2014)
[22] Baldoni, V., et al.: A user's guide for latte integrale v1. 7.1. Optimization 22, 2 (2014)
[23] Belle, V., van den Broeck, G., Passerini, A.: Component caching in hybrid domains with piecewise polynomial densities. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence, pp. 3369–3375 (2016)
[24] Morettin, P., et al. Efficient weighted model integration via SMT-based predicate abstraction. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), pp. 720–728 (2017)
[25] Vergari, A., Di Mauro, N., Esposito, F.: Simplifying, regularizing and strengthening sum-product network structure learning. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 343–358 (2015)
[26] Molina, A., Natarajan, S., Kersting, K.: Poisson sum-product networks: A deep architecture for tractable multivariate poisson distributions. In: Proceedings of 26th International Joint Conference on Artificial Intelligence (IJCAI), pp. 2357–2363 (2017)
[27] Baldoni, V., et al.: How to integrate a polynomial over a simplex. Mathematics of Computation 80(273), 297–325 (2011)
[28] Molina, A., et al.: Mixed sum-product networks: A deep architecture for hybrid domains. In: The 32nd AAAI Conference on Artificial Intelligence (AAAI-18), pp. 1–8 (2018)
[29] Gebelein, H.: Das statistische problem der korrelation als variations-und eigenwertproblem und sein zusammen hang mit der ausgleichsrechnung. ZAMM-Journal of Applied Mathematics and Mechanics/ Zeitschrift für Ange-wandte Mathematik und Mechanik 21(6), 364–379 (1941)

ChinaXiv合作期刊

[30] Kulessa, M.: Model-based approximate query processing. arXiv preprint arXiv: 1811.06224 (2018)
[31] Vergari, A., et al.: Automatic Bayesian density analysis. In: The 33th AAAI Conference on Artificial Intelligence (AAAI-19), pp. 5207–5215 (2019)
[32] Murphy, K.P.: A variational approximation for Bayesian networks with discrete and continuous latent variables. In: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99), pp. 457–466 (1999)
[33] Lauritzen, S.L., Jensen, F.: Stable local computation with conditional Gaussian distributions. Statistics and Computing 11(2), 191–203 (2001)
[34] Nitti, D., et al.: Learning the structure of dynamic hybrid relational models. In: The 22nd European Conference on Artificial Intelligence (ECAI), pp. 1283–1290 (2016)
[35] Ravkic I., Ramon, J., Davis, J.: Learning relational dependency networks in hybrid domains. Machine Learning 100(2–3), 217–254 (2015)
[36] Peharz, R., et al.: On theoretical properties of sum-product networks. In: Proceedings of the 18th International Conference on Artificial Intelligence and Statistics, pp. 744–752 (2015)
[37] Trapp, M., et al.: Bayesian learning of sum-product networks. In: The Neural Information Processing Systems Conference (NIPS 2019), pp. 1–12 (2019)
[38] Peharz, R., et al.: Random sum-product networks: A simple and effective approach to probabilistic deep learning. In: The 35th Conference on Uncertainty in Artificial Intelligence (UAI 2019), pp. 334–344 (2020)
[39] Peharz, R., et al.: Einsum networks: Fast and scalable learning of tractable probabilistic circuits. arXiv preprint arXiv: 2004.06231 (2020)
[40] Belle, V., van den Broeck, G., Passerini, A.: Hashing-based approximate probabilistic inference in hybrid do mains. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence, pp. 4115–4119 (2016)
[41] Zeng, Z., van den Broeck, G.: Efficient search-based weighted model integration. arXiv preprint arXiv: 1903.05334 (2019)
[42] Kolb, S., Dos Martires, P.Z., De Raedt, L.: How to exploit structure while solving weighted model integration problems. In: The Conference on Uncertainty in Artificial Intelligence, pp. 1–11 (2019)
[43] Chistikov, D., Dimitrova, R., Majumdar, R.: Approximate counting in SMT and value estimation for probabilistic programs. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 320–334 (2015)
[44] Barrett, C.W., et al.: Satisfiability modulo theories. In: Biere, A., et al. (eds.) Handbook of Satisfiability, pp. 825–885. IOS Press, Amsterdam (2009)
[45] Shenoy, P.P.: Two issues in using mixtures of polynomials for inference in hybrid Bayesian networks. International Journal of Approximate Reasoning 53(5), 847–866 (2012)
[46] Bekker, J., et al.: Tractable learning for complex probability queries. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, pp. 2242–2250 (2015)
[47] Dougherty, J., et al.: Supervised and unsupervised discretization of continuous features, Machine learning. In: Proceedings of the Twelfth International Conference, pp. 194–202 (1995)
[48] Speichert, S.: Learning Hybrid Relational Rules with Piecewise Polynomial Weight Functions for Probabilistic Logic Programming, Master's thesis, University of Edinburgh (2017)
[49] Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics 6(2), 461–464 (1978)
[50] Lopez-Cruz, P.L., Bielza, C., Larrañaga, P.: Learning mixtures of polynomials of multidimensional probability densities from data using B-spline interpolation. International Journal of Approximate Reasoning 55(4), 989–1010 (2014)
[51] Dheeru, D., Taniskidou, E.K.: UCI machine learning repository. Available at: http://archive.ics.uci.edu/ml/about.html. Accessed 10 October 2020

chinaXiv:202211.00326v1

ChinaXiv合作期刊

chinaXiv:202211.00326v1

AUTHOR BIOGRAPHY

Andreas Bueff is a postgraduate research student in the Artificial Intelligence and its Applications Institute at the School of Informatics, University of Edinburgh. He is interested in explainable AI as a platform for exploring unsupervised and reinforcement learning methods, research into tractable learning of probabilistic graphical models across continuous and discrete data domains, and the interpretability of relational reinforcement learning agents in model-free learning environments, including the use of counterfactuals for better agent understanding. As part of his research, he has collaborated with the University of Edinburgh, Business School in conjunction with Nationwide in stress scenario testing of credit risk models.
ORCID: 0000-0002-7538-0699

Stefanie Speichert is a final year PhD student at the Institute for Perception, Action and Behavior at the School of Informatics, University of Edinburgh. She is interested in interpretable machine learning methods with a particular focus on bridging the gap between traditional probabilistic planners and neural networks. Her previous work on probabilistic logic programming in hybrid discrete-continuous domains has received the best student paper award (long paper track) at the International Conference on Inductive Logic Programming (ILP) in 2019. She was further able to apply her knowledge to the medical and self-driving domain through two internships at IBM Research and Lyft's Self-Driving Research lab, respectively. When she is not working she loves to engage in science communication which has led her to co-found and lead two of the largest Edinburgh-based AI public outreach groups to democratize AI knowledge to students and professionals alike.
ORCID: 0000-0003-1122-0685

ChinaXiv合作期刊

Vaishak Belle is a Chancellor's Fellow and Faculty at the School of Informatics, University of Edinburgh, an Alan Turing Institute Faculty Fellow, a Royal Society University Research Fellow, and a member of the RSE (Royal Society of Edinburgh) Young Academy of Scotland. At the University of Edinburgh, he directs a research lab on artificial intelligence, specializing in the unification of symbolic systems and machine learning, with a recent emphasis on explainability and ethics. He has given research seminars at numerous academic institutions, tutorials at AI conferences, and talks at venues such as Ars Electronica and the Samsung AI Forum. He has co-authored over 50 scientific articles on AI, at venues such as IJCAI, UAI, AAAI, MLJ, AIJ, JAIR, AAMAS, and along with his co-authors, he has won the Microsoft best paper award at UAI, the Machine learning journal best student paper award at ECML-PKDD, and the Machine learning journal best student paper award at ILP. In 2014, he received a silver medal by the Kurt Goedel Society. Recently, he has consulted with major banks on explainable AI and its impact in financial institutions.

ORCID: 0000-0001-5573-8465

chinaXiv:202211.00326v1